A Network Intrusion Detection System Using Ensemble Machine Learning

Aklil Zenebe Kiflay
Cyber Security Center
University of Hertfordshire
Hatfield, United Kingdom
a.z.kiflay@herts.ac.uk

Athanasios Tsokanos
Cyber Security Center
University of Hertfordshire
Hatfield, United Kingdom
a.tsokanos@herts.ac.uk

Raimund Kirner
Cyber Security Center
University of Hertfordshire
Hatfield, United Kingdom
r.kirner@herts.ac.uk

2021 International Carnahan Conference on Security Technology (ICCST) | 978-1-6654-9988-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCST49569.2021.9717397

***Abstract*—The type and number of cyber-attacks on data networks have been increasing. As networks grow, the importance of Network Intrusion Detection Systems (NIDS) in monitoring cyber threats has also increased. One of the challenges in NIDS is the high number of alerts the systems generate, and the overwhelming effect that alerts have on security operations. To process alerts efficiently, NIDS can be designed to include Machine Learning (ML) capabilities. In the literature, various NIDS architectures that use ML approaches have been proposed. However, high false alarm rates continue to be challenges to most NID systems. In this paper, we present a NIDS that uses ensemble ML in order to improve the performance of attack detection and to decrease the rate of false alarms. To this end, we combine four ensemble ML classifiers – (Random Forest, AdaBoost, XGBoost and Gradient boosting decision tree) using a soft voting scheme.**

Keywords—ensemble machine learning, network intrusion detection, security monitoring.

I. INTRODUCTION

Modern computer networks use intrusion detection systems (IDS) to monitor and analyze cyber-attacks. As networks grow in size and complexity, intrusion detection and prevention systems (IDS/IPS) have become crucial elements of the overall security landscape. IDSs monitor cyber-attacks and security policy violations at different levels in a network. By enabling security experts to discover potential threats quickly, IDS speeds up detection of cyber-attacks, thereby shortening the dwell time of cyber threats inside a network [1], [2], [3].

The data processed by an IDS can be collected from end systems, network devices, or both, depending on the intended focus of the security monitoring. In terms of data sources, IDSs are classified into host-based, network-based, and hybrid [4]. A Network-based IDS (NIDS) examines traffic that passes through the network by packet filtering [5]. In contrast, a host-based IDS (HIDS) utilizes access logs from end systems [6], [7], [8], [9]. Fundamentally, an NIDS has a broader security monitoring coverage than an HIDS because an NIDS can track intrusions into multiple hosts in a network and intermediate devices. Depending on site-specific design considerations, a given network can have multiple instances of an NIDS, as well as hybrid implementations of an NIDS and an HIDS to monitor security events at different network segments and end systems. In terms of detection techniques, an NIDS can use signature based, anomaly-based, or hybrid approaches to detect attacks. [10], [11], [12].

Fig. 1. shows an intrusion detection usage scenario that consists of a Network IDS, a Host IDS and a Security Incident and Event Manager (SIEM) system. In this scenario, the network IDS takes raw traffic at an entry point to the network, and generates alerts when it finds suspicious content. The alerts generated by an NIDS can be passed directly to human experts for analysis and action. Alternatively, they can be directed to a SIEM system, which integrates NIDS alerts with data from other sources such as Host IDS and logging systems, in order to provide situational awareness and a complete view of an organization's information security [13]. Previous research has shown that excessive NIDS alerts, many of which can be false alarms, cause a common problem known as threat-alert fatigue [14], [15]. In a situation where threat-alert fatigue prevails, the cybersecurity expert spends unduly time investigating too many false alarms instead of responding to real attacks.

Unlike a human expert, who can be overwhelmed by a threat-alert fatigue, the more data an ML-based NIDS receives, the more accurate decision it can make in detecting attack traffic. The use of machine learning, in both signature-based and anomaly-based NIDS, has been demonstrated with varying degrees of performance [16]. However, high rate of false alarms continues to hamper practical adoptions of ML-based NIDS systems. The effect of false positives and false negatives in other domains, such as spam filtering and recommendation systems, can be tolerated. Because false alarms in such systems may not be fatal to the overall operation of the systems. In contrast, the cost of errors in NIDS systems is high because of the increasingly adversarial environment where attackers are always looking for security holes and trying to exploit them [17].

In this paper, we propose an NIDS that uses ensemble machine learning in order to improve the performance of the attack detection rate and decrease the rate of false alarms. We use machine learning to process network traffic and detect potential cyber threats in such a way that the flood of false alerts that reach the security expert can be minimized, which in turn enables the expert to focus on real network threats. In the proposed NIDS, we train and test the random forest [18], AdaBoost [19], XGBoost [20], and gradient boosting decision tree (GBDT) [21] classifiers, each of which is an ensemble learner based on

978-1-6654-9988-0/21/$31.00 ©2021 IEEE

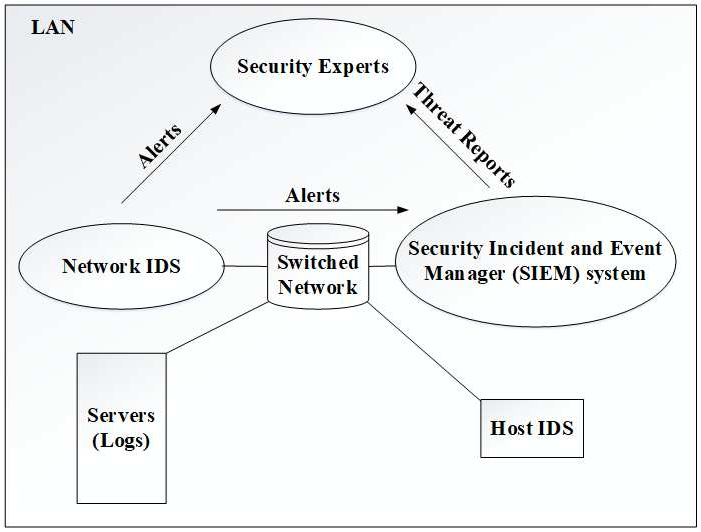

Fig. 1. NIDS and a common intrusion detection workflow

decision trees. An ensemble model aggregates base estimators in such a way that it takes advantage of the best performances from the individual estimators. In the NIDS domain, various combinations of ensemble learning approaches have been tested to improve the detection accuracy of attacks [22], [23], [24]. But to the best of our knowledge, there is no existing NIDS using the proposed combination of methodology.

The rest of the paper is organized as follows. Section II provides a background on the use of machine learning in network intrusion detection. Related works are discussed in section III and we present our NIDS in section IV. Finally, section V concludes the paper.

II. BACKGROUND

Although there has been a lot of research on using machine learning for intrusion detection, there are gaps in terms of successful practical deployments. For example, Sommer and Paxson argued that the field of network intrusion detection presents unique challenges [17]. The authors concluded that the challenges are inherent to the field, and may not exist in other domains where practical deployments of machine learning based systems have been largely successful. Some of these challenges include lack of stable notion of normality due to network variability, diversity of network traffic, lack of suitable public IDS datasets, and an adversarial environment where attackers actively look for security holes.

In IDS research, it is well established that current intrusion detection systems have limitations. Research on popular open source network IDS systems such as Snort and Suricata has shown that errors occur during detection of attacks [1], [25], [11]. Snort and Suricata depend on predefined rules to identify malicious network traffic [12], [25]. By using attack signature databases, they look for intrusive incoming or outgoing traffic. One of the drawbacks of this detection approach is that it takes time for signatures of new attacks to be added to the IDS database. To do that, network security experts need to study the behavior of new attacks in a controlled environment, and then manually write the rules for detecting them. Since new vulnerabilities are discovered regularly, following which attackers develop exploits to take advantage of the vulnerabilities, a manual IDS rule creation can be cumbersome, time consuming, and ineffective. Additionally, given the sheer size of data in modern networks, searching for attacks can be almost as daunting as finding a needle in a haystack.

To address these challenges, the use of machine learning in the detection of cyber-attacks has gained popularity within the research community as well as the industry [26], [27], [28], [29].

In supervised machine learning, ensemble learning is a term used to describe an aggregation of base learners in such a way that the ensemble's overall prediction performance is better than that of a single base learner [30]. Ensemble methods have better predictive performance than single learners partly because ensemble learning minimizes overfitting and handles class imbalance better [31]. Broadly, the majority of ensemble learners that are based on decision trees belong to either bagging or boosting [32].

A boosting ensemble turns a set of weak base learners to a strong learner with higher accuracy, by making each weak learner concentrate on different aspects of the data distribution. To pick up errors made by each weak learner, boosting ensemble trains the weak learners sequentially, by making later learners focus on errors made by previous learners. This enables each base learner of the boosting ensemble to focus more on aspects of the data that the other learners got wrong. Boosting results in a more accurate prediction when the results are combined [19]. AdaBoost [19] is one of the earliest boosting learners.

In contrast, a bagging ensemble trains the same algorithm with different samples of the training data. The final predicted output is the average of the sub-models on different subsets of the training data. The bagging ensemble aggregates the outputs of the base learners using voting for classification problems and averaging for regression problems. The most popular bagging ensemble models are bagged decision tree and random forest [32].

A. *Random Forest*

Random forest (RF), first proposed by Breiman [18], is one of the most widely used ensemble methods. It constructs its constituent trees with the aim of reducing the correlation between individual decision trees. Random forest is an extension of bagged decision tree [33] with the introduction of randomized attribute selection. Decision trees in a random forest are created based on a subset of the data attributes. It is important to note that the performance gains in a random forest come from the randomness in the attribute selection process, not from the split points in the decision trees of the selected attributes.

B. *Adaptive Boosting*

As a boosting learner, AdaBoost classifier, introduced by Freund et al. [19], creates sequential classifiers on the same dataset in such a way that later classifiers work to improve on the classification errors of the previous ones. To do this, AdaBoost adjusts weights of incorrectly classified instances from a previous learner so that subsequent learners focus on the difficult cases. AdaBoost-SAMME [34] is an implementation of the AdaBoost algorithm.

C. Gradient Boosting Decision Tree

Gradient boosting decision tree (GBDT) was first introduced by Friedman et al. [21]. Similar to other boosting methods such as AdaBoost, gradient tree boosting is made up of multiple decision trees to create a strong learner. Gradient boosting works by building a number of decision trees with the aim of reducing the classification error. It uses differentiable loss functions to improve the accuracy of its predictions.

D. Extreme Gradient Boosting

Extreme gradient boosting (XGBoost) [20] is a more accurate and scalable implementation of the original gradient boosting proposed by Friedman et al. [21]. In general, XGBoost has superior performance over that of gradient boosting because XGBoost supports parallel computing, cache awareness, a built-in regularisation technique to reduce overfitting, and tree optimization by a split-finding algorithm.

III. RELATED WORK

Hsu et al. designed an ensemble learning approach for NIDS using random forest, support vector machine, and auto-encoder models [35]. By carrying out experiments on a three-part anomaly-based intrusion detection engine, the authors demonstrated that ensemble models can increase classification accuracy and decrease false alarm rates. Gao et al. proposed an ensemble learning approach in which they used decision tree, random forest, k-nearest neighbour, and deep neural network as base classifiers [36]. With their adaptive ensemble model, they reported increased detection accuracy.

Verma and Ranga presented an ensemble model for NIDS which consisted of boosted trees, bagged trees, RUSBoosted trees and subspace discriminant [37]. In a network intrusion detection system for Internet of Things, the authors concluded that an ensemble approach can improve detection performance. Another ensemble model was proposed by Ludwig to classify different attacks [38]. In this ensemble method, which consisted of autoencoder, deep belief network, deep neural network, and an extreme learning machine, the author reported performance enhancements in terms of precision, recall, and F1 score.

Various other NIDS models with ensemble learning were proposed by Zhou et al. [22], Devan et al. [23], and Zhang et al. [24]. In a similar way to the work by Zhou et al. [22], we use a soft voting scheme based on the probability distributions from the base learners to differentiate attack from normal traffic. The difference from our work is that Zhou et al. [22] used C4.5, random forest, and forest penalizing attributes in their ensemble model.

IV. PROPOSED NIDS

In this paper, we present an NIDS that has three major components, namely Packet Capture Unit (PCU), Data Processing Unit (DPU), and Classifier Engine (CE). The PCU extracts attribute values from network traffic. The DPU takes a mix of quantitative and qualitative data from PCU and generates normalized numeric data. First, the DPU converts qualitative values to number representations to make the data suitable for processing by machine learning algorithms. Next, since outlier data values have been known to create problems for ML algorithms, the DPU normalizes the data. Finally, the Classifier Engine (CE) is responsible for classifying network traffic samples into normal or attack traffic.

Fig. 2. shows the three major components of the proposed NIDS, and how information flows from a network data capture to a SIEM system, which provides a point of analysis for security experts who can take action based on the gathered intelligence.

A. Packet Capture Unit

The purpose of this module is to capture network traffic for security monitoring. The packet capture unit (PCU) uses network sniffing tools such as Wireshark, Tcpdump, and Zeek IDS (formerly Bro IDS) to capture raw network packets for security monitoring. In principle, the PCU in our proposed NIDS is similar to traffic sniffing procedures used by Moustafa et al. to prepare the UNSW-NB15 [39] dataset, and by the authors of the KDDCup99 [40] dataset from which the NSL-KDD [41] was derived. Therefore, we decided to use UNSW-NB15 and NSL-KDD public IDS datasets instead of our own network traffic capture. Because doing so enables us to make comparisons with results of similar works reported in the literature. When selecting which public IDS datasets to work with, we followed dataset evaluation frameworks proposed by Gharib et al. and Ring et al. [42], [43].

We use the full list of traffic attributes described by Moustafa et al. [39] and Tavallaee et al. [41]. Some examples of these network traffic attributes are protocol type, service type, flag, source bytes, destination bytes, source TTL, destination TTL, source window, destination window, TCP round trip time, SYN and ACK. The traffic attribute values of the datasets are then forwarded to the Data Pre-processing Unit to be converted to a machine learning friendly format.

B. Data Pre-processing Unit

The main function of the data pre-processing unit (DPU) is to prepare the data for the machine learning algorithms used in the classifier engine. Data pre-processing is necessary because network traffic attributes contain a mix of numeric and categorical attribute values. First, the DPU converts categorical attribute values, such as protocol type (e.g. TCP, UDP, etc.), service type (HTTP, DNS, SMTP, DHCP, etc.), into numeric representations using one-hot encoding [44]. Rare outlier values can impact the classification performance of the ML algorithms even after all attributes have been converted to numbers. Therefore, it is important to convert all attribute values so that they fall within a specific range to avoid problems caused due to varying measurement scales. To this end, for any attribute value x, the DPU calculates its normalized value x_n as shown in (1), where $\mu(x)$ and $\delta(x)$ are the mean and the standard deviation, respectively.

$$x_n = \frac{x - \mu(x)}{\delta(x)} \quad (1)$$

C. Classifier Engine

The classifier engine (CE) is the core part of the NIDS since it classifies network traffic into benign or attack. In principle, the classifier engine can use any machine learning classifier algorithm. Our classifier engine uses random forest, AdaBoost, XGBoost, and gradient boosting decision tree (GBDT) as

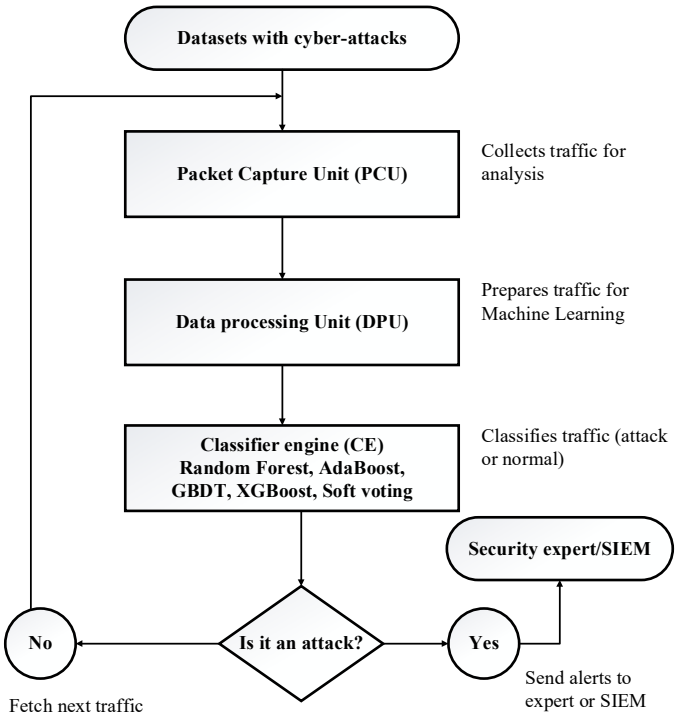

Fig. 2. The proposed NIDS

classifiers. Additionally, we integrate these four ensemble methods using a soft voting classifier.

The predictive power of ensemble learning is usually better than that of its constituent methods. For example, the random forest classifier is created from a group of decision trees, each trained on randomly selected subset of the data. By leveraging the outcomes of its constituent trees, random forest predicts a class that is voted by majority of the decision trees. In order to combine the classification outputs of random forest, AdaBoost, GBDT, and XGBoost, we use scikitlearn's voting classifier [45] with a soft voting scheme.

D. Voting classifier

On a subset of the data, some base learners of an ensemble have superior performance than others. Rather than selecting a single learner, combination methods can be applied on the base learners to achieve a stronger generalization. Some examples of combination methods are averaging, majority voting, plurality voting, weighted voting, and soft voting. Dietterich [31] concluded that the generalization benefits are possible because a combination method addresses statistical, computational and representational problems of the training dataset, and of the base learners.

For classification problems, such as intrusion detection, voting is one of the most appropriate combination methods. The outputs of a base classifier can be crisp labels or class probabilities [32]. Soft voting is an ensemble combination method based on class probabilities. Given a set of base classifiers, the task of a voting classifier is to combine the outputs of the base classifiers in such a way that the prediction accuracy of the ensemble is maximized. For heterogeneous ensembles, voting schemes such as majority voting and weighted voting are recommended, whereas a soft voting scheme can be used in homogeneous ensembles [32].

For a given n individual classifiers $\{h_1, \dots, h_n\}$, and a traffic sample x, soft voting combines traffic class labels from $|h_i|_{i\in(1,\dots,n)}$ to predict the final outcome from the possible labels of $\{0, 1\}$ in the case of binary classification. The output of each classifier h_i is given as a two-dimensional label vector $[h_i^0(x), h_i^1(x)]^T$ where $h_i^0(x)$ is the output of $h_i(x)$ for predicted traffic label 0 (normal), whereas $h_i^1(x)$ is the output of $h_i(x)$ for predicted traffic label 1 (attack).

Based on the output from each classifier, the values of $h_i^0(x)$ and $h_i^1(x)$ can be in the form of a crisp label or of a class probability. For a crisp label output,

$$h_i^j(x)=\begin{cases}1, & \text{if } h_i^j(x) \text{ predicts x is attack traffic}\\ 0, & \text{if } h_i^j(x) \text{ predicts x is normal traffic}\end{cases}$$

where $j = 0, 1$.

In contrast, when the outputs of the individual classifiers are in the form of class probabilities, they denote the confidence level of classifying the sample as normal or attack. In this case, each value of $h_i^j(x) \in [0, 1]$ is the posterior probability $P(c_j|x)$, for a class label $c_j, j = 0, 1$.

In order to make the final decision on the predicted traffic label, majority voting, plurality voting, and weighted voting can be used when the results of the individual classifiers are crisp class labels. But when class probability outputs are available from the individual classifiers, soft voting is generally a better choice [32]. Furthermore, in cases where the individual classifiers are treated equally by the soft voting scheme, the aggregated and final class prediction $H^j(x)$ is calculated by (2).

$$H^j(x)= \frac{1}{n} \bullet \sum_{i=1}^{n} h_i^j(x) \qquad (2)$$

Fig. 3. shows how traffic samples are processed using four classifiers in parallel in our proposed NIDS. It depicts how four traffic class probabilities from each classifier are forwarded to

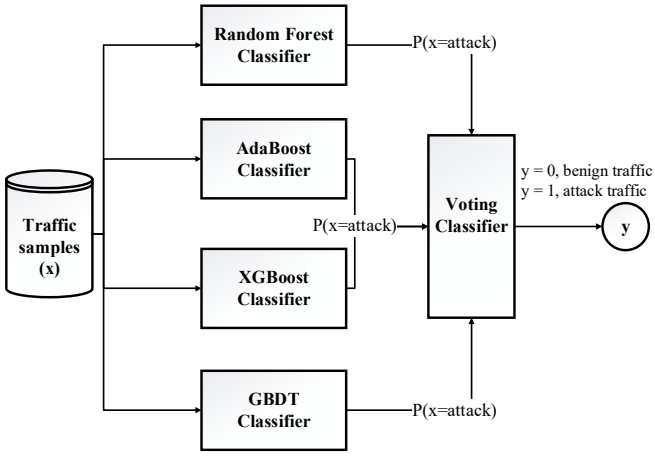

Fig. 3 Soft voting classifier used by the proposed NIDS

the voting classifier, which then determines the final traffic label of the sample as attack or normal. The performance of the proposed NIDS was evaluated using the *NSL-KDD* and *UNSW-NB15 IDS* datasets. The results show that the proposed NIDS has potential to improve the accuracy of cyber-attack detection, and to minimize the rate of false alarms.

V. CONCLUSION

In this paper, we presented a new Network Intrusion Detection System (NIDS) that uses ensemble machine learning to classify network traffic as attack or benign. Using the *NSL-KDD* and *UNSW-NB15* IDS datasets, the proposed ensemble NIDS was evaluated for binary traffic classification (benign or attack) with the full set of traffic attributes. The results show that the proposed NIDS has potential to improve the accuracy of cyber-attack detection, and to minimize the rate of false alarms.

In the future, we intend to test the proposed NIDS with a selected subset of the traffic attributes and for the identification of specific types of attacks (multiclass classification).

REFERENCES

[1] M. Roesch and others, "Snort: Lightweight intrusion detection for networks.," in Lisa, 1999, vol. 99, pp. 229–238.

[2] V. Paxson, "Bro: a system for detecting network intruders in real-time," Comput. networks, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[3] V. Paxson, S. Campbell, J. Lee, and others, "Bro intrusion detection system," 2006.

[4] D. E. Denning, "An intrusion-detection model," IEEE Trans. Softw. Eng., no. 2, pp. 222–232, 1987.

[5] V. Jacobson, C. Leres, and S. McCanne, "The tcpdump manual page," Lawrence Berkeley Lab. Berkeley, CA, vol. 143, 1989, [Online]. Available: https://www.tcpdump.org/.

[6] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," ACM Comput. Surv., vol. 51, no. 5, pp. 1–36, 2018.

[7] Y. Lin, Y. Zhang, and Y. Ou, "The design and implementation of host-based intrusion detection system," in 2010 third international symposium on intelligent information technology and security informatics, 2010, pp. 595–598.

[8] L. Vokorokos and A. Baláž, "Host-based intrusion detection system," in 2010 IEEE 14th International Conference on Intelligent Engineering Systems, 2010, pp. 43–47.

[9] S. N. Chari and P.-C. Cheng, "BlueBox: A policy-driven, host-based intrusion detection system," ACM Trans. Inf. Syst. Secur., vol. 6, no. 2, pp. 173–200, 2003.

[10] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," Telecommun. Syst., vol. 70, no. 3, pp. 447–489, 2019, doi: 10.1007/s11235-018-0475-8.

[11] A. Alhomoud, R. Munir, J. P. Disso, I. Awan, and A. Al-Dhelaan, "Performance evaluation study of intrusion detection systems," Procedia Comput. Sci., vol. 5, pp. 173–180, 2011.

[12] B. R. Murphy, "Comparing the Performance of Intrusion Detection Systems: Snort and Suricata," Colorado Technical University, 2019.

[13] S. Bhatt, P. K. Manadhata, and L. Zomlot, "The operational role of security information and event management systems," IEEE Secur. Priv., vol. 12, no. 5, pp. 35–41, 2014, doi: 10.1109/MSP.2014.103.

[14] W. U. Hassan et al., "Nodoze: Combatting threat alert fatigue with automated provenance triage," 2019.

[15] M. E. Aminanto, T. Ban, R. Isawa, T. Takahashi, and D. Inoue, "Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis," IEEE Access, vol. 8, pp. 217977–217986, 2020.

[16] R. Chapaneri and S. Shah, "A comprehensive survey of machine learning-based network intrusion detection," in Smart Innovation, Systems and Technologies, 2019, vol. 104, pp. 345–356, doi: 10.1007/978-981-13-1921-1_35.

[17] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in In proceedings of the 2010 IEEE symposium on security and privacy, 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[18] L. Breiman, "Random forests," Mach. Learn., vol. 45, no. 1, pp. 5–32, 2001.

[19] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, 1997.

[20] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[21] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Ann. Stat., pp. 1189–1232, 2001.

[22] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," Comput. Networks, vol. 174, p. 107247, 2020.

[23] P. Devan and N. Khare, "An efficient XGBoost–DNN-based classification model for network intrusion detection system," Neural Comput. Appl., pp. 1–16, 2020.

[24] Z. Zhang, P. Chirupphapa, H. Esaki, and H. Ochiai, "XGBoosted Misuse Detection in LAN-Internal Traffic Dataset," in 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), 2020, pp. 1–6.

[25] E. Albin and N. C. Rowe, "A realistic experimental comparison of the Suricata and Snort intrusion-detection systems," in 2012 26th International Conference on Advanced Information Networking and Applications Workshops, 2012, pp. 122–127, doi: 10.1109/WAINA.2012.29.

[26] J. Martínez Torres, C. Iglesias Comesaña, P. J. García-Nieto, J. M. Torres, C. I. Comesaña, and P. J. Garc\'\ia-Nieto, "Review: machine learning techniques applied to cybersecurity," Int. J. Mach. Learn. Cybern., vol. 10, no. 10, pp. 2823–2836, 2019, doi: 10.1007/s13042-018-00906-1.

[27] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in 2018 10th International Conference on Cyber Conflict (CyCon), 2018, vol. 2018-May, pp. 371–390, doi: 10.23919/CYCON.2018.8405026.

[28] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Commun. Surv. Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[29] N. Alqudah and Q. Yaseen, "Machine Learning for Traffic Analysis: A Review," Procedia Comput. Sci., vol. 170, pp. 911–916, 2020, doi: 10.1016/j.procs.2020.03.111.

[30] O. Sagi and L. Rokach, "Ensemble learning: A survey," Wiley Interdiscip. Rev. Data Min. Knowl. Discov., vol. 8, no. 4, p. e1249, 2018.

[31] T. G. Dietterich, "Ensemble methods in machine learning," in International workshop on multiple classifier systems, 2000, pp. 1–15.

[32] Z.-H. Zhou, Ensemble Methods. CRC press, 2012.

[33] L. Breiman, "Bagging predictors," Mach. Learn., vol. 24, no. 2, pp. 123–140, 1996.

[34] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," Stat. Interface, vol. 2, no. 3, pp. 349–360, 2009.

[35] Y.-F. Hsu, Z. He, Y. Tarutani, and M. Matsuoka, "Toward an online network intrusion detection system based on ensemble learning," in 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), 2019, pp. 174–178.

[36] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An Adaptive Ensemble Machine Learning Model for Intrusion Detection," IEEE Access, vol. 7, pp. 82512–82521, 2019, doi: 10.1109/ACCESS.2019.2923640.

[37] A. Verma and V. Ranga, "ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things," in 2019 4th International conference on Internet of Things: Smart innovation and usages (IoT-SIU), 2019, pp. 1–6.

[38] S. A. Ludwig, “Intrusion detection of multiple attack classes using a deep neural net ensemble,” in 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017, pp. 1–7.

[39] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in 2015 Military Communications and Information Systems Conference, MilCIS 2015 - Proceedings, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[40] K. D. D. Cup, “Dataset,” available Follow. website http//kdd. ics. uci. edu/databases/kddcup99/kddcup99. html, vol. 72, p. 15, 1999.

[41] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.

[42] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “An evaluation framework for intrusion detection dataset,” in 2016 International Conference on Information Science and Security (ICISS), 2016, pp. 1–6.

[43] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” Comput. Secur., vol. 86, pp. 147–167, 2019, doi: 10.1016/j.cose.2019.06.005.

[44] P. Cerda, G. Varoquaux, and B. Kégl, “Similarity encoding for learning with dirty categorical variables,” Mach. Learn., vol. 107, no. 8, pp. 1477–1494, 2018.

[45] Scikit-Learn, “Voting Classifier,” 2021. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html# (accessed Feb. 01, 2021).